A RECONFIGURABLE CONTROL STRUCTURE FOR CPUS AND METHOD OF OPERATING SAME

Field of the Invention

The present invention relates to control units for central processing units (CPUs).

Description of the Related Art

In a CPU, it is generally possible to distinguish two sections referred to as "paths:"

the path followed by the data (data path), which involves the arithmetic-logic unit (ALU), registers and buses; and

the path followed by the control signals (control path) to the devices of the data path.

In the context of the control path, selection signals (for operations on the ALU, selection of registers) and timing signals (clock, enabling signals) can moreover be distinguished.

In the above context of application, currently designated by control unit (CU) is a finite state machine designed for coordinating and managing the signal-activation timing sequences according to the types of processing required by the CPU. For each instruction, the control unit guides the CPU through a sequence of control states. In each control state, the control unit sends a set of signals which cause execution of the corresponding micro-operations.

The general scheme of such a control unit is represented in Figure 1, where the reference CU designates precisely said control unit. Operation of the control unit CU is clocked by a clock signal CLK. The reference F designates, as a whole, the so-called state flags and conditioning flags, whereas the references CS1, CS2 and SB designate, respectively:

the set of internal control signals to the CPU (not illustrated in the diagram of Figure 1);

the control signals on the system bus; and

the signals coming form the system bus;

The reference IR designates the register where the current instruction is stored.

Again for purposes of general orientation, it is recalled that a finite state machine is a Moore state machine if the combinatorial logic which generates the outputs is only a function of the current state, whereas it is a Mealy state machine if the combinatorial logic that generates the outputs is a function both of the current state and of the inputs.

The solutions adopted for obtaining control units of the type considered previously amount to two fundamental models, *i.e.*:

wired-logic solutions; and

microprogrammed-logic solutions.

The general scheme of a wired-logic control unit is represented in Figure 2, where the references IR, F, SB and CLK have the same meaning to which reference has already been made previously in the description of Figure 1.

The core of the control unit represented in Figure 2 is made up of an encoder block EB basically consisting of a combinatorial-signal generator. The generator in question is designed to receive at input the instructions (decoded by an appropriate instruction decoder ID), the flags F, the control signals SB coming from the system bus SB, as well as the clock signal CLK processed by a module SC designed to function as step counter and decoder, the purpose being to generate at output control signals CS generated according to a typical parallel format.

When the solution described in Figure 2 is used, once the finite state machine that characterizes the wired-logic control unit has been defined, there follows a step of synthesis of the combinatorial networks required for calculating

the outputs CS and the next state of the control unit, starting from the current state and the inputs. The block EB, therefore, basically consists of a circuit made up of flip-flops connected via combinatorial networks, *i.e.*, via more or less complex networks of logic gates.

One of the major critical elements in a structure of this type is linked to the rigidity of the design. A modification of the control policies necessarily affects the design and dimensions of the finite state machine, with a marked incidence on production times and costs thereof.

The diagram of Figure 3 represents, instead, the basic structure of a microprogrammed-logic control unit. Also in this case, the references IR, F, SB, CLK, and CS indicate the same entities already designated in the same way in Figure 1 and/or Figure 2.

In the solution represented in Figure 3, the current instruction contained in the register IR and the flags F converge towards an address generator AG. The address generator AG operates under the control of a sequencer S, which is clocked by the clock signal CLK and at which there arrive the signals from the system bus SB. The addresses generated by the generator AG reach a microprocessor MPC, the operation of which evolves under the control of an incremental signal INCR coming from the sequencer R. The microprocessor MPC co-operates with a memory MCS, commonly referred to as control memory, in which a microprogram is stored, which defines the sequences of the control signals of the finite state machine. The output from the control memory CS drives a further circuit CB/D, which functions as control buffer/decoder. The circuit CB/D generates the control signals CS that are to be sent to the CPU. In the diagram of Figure 3, there is also visible an internal address bus, designated by IAB, as well as a line EEF.

The module CB/D transmits on the bus IAB to the generator AG the current state of the control unit, whilst, on the line designated by EEF, it transmits

internal control signals indicating the possible end of the instruction END or the end of the fetch step END FETCH or the end of the branch BRANCH.

The versatility of the solution illustrated in Figure 3 appears evident once it is decided to modify the control policies. Leaving unaltered the combinatorial structure that supervises operation of the finite state machine, by adopting the solution represented in Figure 3, it is sufficient to act on a microprogram stored in memory to obtain the necessary variations. Furthermore, the design of the circuitry part and the design of the instruction set may be performed in parallel, with savings both in terms of time and in terms of costs.

BRIEF SUMMARY OF THE INVENTION

An embodiment of the present invention provides a reconfigurable control structure which enables amplification and/or modification of the instruction set of a CPU by appropriately programming a memory in which the sequences of states defining the control signals required for execution of a given instruction are stored.

The control structure includes a programmable unit such as will enable the user to define new executable instructions and/or to redefine the basic implemented instructions.

Such an architecture means that the corresponding system will assume the characteristics of an open system: the user can, in fact, define the instruction set that he must use or that a particular application requires. A reconfigurability of this sort renders the microprocessor flexible, enabling the dimensions thereof to be contained, at the same time as the said microprocessor will be able to perform fully the instructions for which it is designed. The control structure thus combines both the advantages of an architecture of a CISC (Complete-Instruction-Set Computer) type, with complex instructions for the required applications, and the advantages of an architecture of the RISC (Reduced-Instruction-Set Computer) type, with a very small control unit.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

The invention will now be described, purely by way of non-limiting example, with reference to the annexed drawings, in which:

Figures 1 to 3, which represent, as a whole, the prior art, have already been described previously;

Figure 4 represents, in the form of a block diagram, a reconfigurable control structure according to the invention; and

Figure 5 is a diagram representing the memory of a control unit comprised in a structure according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

In the diagram of Figure 4, the references UC0 and UC1 designate two control units designed to operate in combination with one another.

The control unit UC0 is basically a wired-logic control unit, hence having the general structure represented in Figure 2.

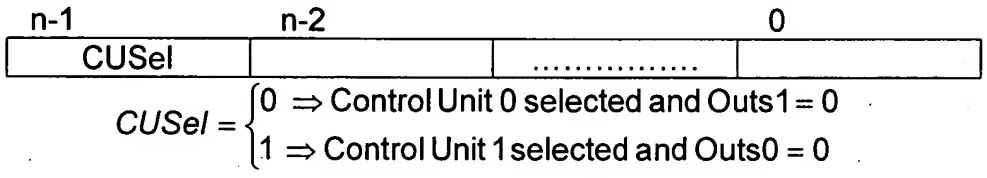
Set alongside the control unit UC0 is the control unit UC1, which corresponds basically to the structure represented in Figure 3, hence to a programmable control unit.

In this connection, it will be appreciated that the indication "Inputs" which appears in Figure 4 also includes the conditioning flags designated by F in Figures 1, 2 and 3. The operating code of the instruction in the solution described here, is designated in Figure 4 by CodOp. Consequently, the register 10 of Figure 4 is, in effect, the analog of the register IR of Figures 1, 2 and 3. The term "Input" has been used rather than "Flag" in order to render the description more general. The input signals "Inputs" (state signals, conditioning signals, etc., hereinafter referred to as "Inputs") – which are assumed as being organized as data in parallel format on k bits – are input into both of the units UC0 and UC1.

It is envisaged that the first unit UC0 will implement the control with reference to a basic instruction set (hence, a set that is the same as or smaller than a typical RISC set).

The second unit UC1, since it is programmable, can instead be used both for implementing the same instruction set as the control unit UC0 (for example, for debugging purposes or troubleshooting purposes) and for implementing new more complex instructions (hence, bringing the performance of the assembly practically up to that of a CISC system).

The use of the solution represented in Figure 4 envisages the construction of an operating code CodOp (in general, on n bits) organized according to the following format:



$$CUSel = \begin{cases} 0 \Rightarrow \text{Control Unit 0 selected and Outs1} = 0 \\ 1 \Rightarrow \text{Control Unit 1 selected and Outs0} = 0 \end{cases}$$

In the format illustrated above, the most significant bit, designated by CUSel, has the task of selecting the unit that generates the control signals for the instructions to be executed.

By means of a register 10 having the function of time memory, the operating code CodOp is sent both to the control unit UC0 and to the control unit UC1.

The n-th bit of the operating code, *i.e.*, CodOp[n-1], which corresponds to the CUSel bit, is sent to a selection circuit 12 designed for driving two multiplexers 14 and 16.

The first multiplexer 14 picks up the outputs of both of the control units UC0, UC1 – outputs designated by CS0 and CS1, respectively – and sends, on the output line of the control unit, designated by 18, a signal CS corresponding to the signal CS0 or CS1 according to the unit (UC0 or UC1) selected as unit that is to generate the control signals for the instruction to be executed.

A second multiplexer 16 selects, according to modalities altogether similar, which signal is to be sent to a state register 20 for identifying the state that is to be considered as current state of the system.

Specifically, the multiplexer 16 sends, to the register 20, a state signal NS (in general organized on j bits) chosen from between two homologous signals NS0 and NS1 generated by the unit UC0 and the unit UC1, respectively.

Both the multiplexer 14 and the multiplexer 16 operate according to the output signal Sel generated by the selection circuit 12.

In particular, the multiplexer 16 causes the signal NS to correspond to the state signal generated by the control unit (UC0 or UC1) which, at the moment, is generating the control signals for the instructions to be executed.

The control unit UC0 functions, in general, as a finite state machine of a traditional type. Starting from an initial idle or inactive state IDLE, the operating code of the instruction is received at input: if the instruction in question is found to form part of the basic instruction set, the unit UC0 passes to the next state, executing the instruction; otherwise, the unit UC0 remains in the idle state IDLE leaving the respective outputs at "0".

Operation of the control unit UC1 is, as a whole, similar, except that its instruction set is appropriately programmed by the control unit UC0 by means of an instruction within the basic instruction set dedicated to said function.

Basically, the solution described here envisages duplication of the control unit in the two units UC0 and UC1. The first unit in question, *i.e.*, the unit UC0, is of the hardwired type, *i.e.*, with a definitively fixed structure, according to the criteria commonly adopted in the prior art. Instead, the unit UC1 is programmable and hence flexible. Programming of the latter unit is performed by the unit UC0 by means of appropriate instructions, in practice with a memory-programming operation. For this reason, the programming instruction is included in th basic instruction set.

The table appearing below reproduces the so-called truth table of the selector module 12.

CodOp[n-1]	CurrState[j-1]	CurrState[j-2]	Sel
0	0	000...0	0
1	0	000...0	1
-	0	≠0	0
-	1	≠0	1

In the table, the four columns represent, respectively:

the value of the most significant bit of CodOp, namely CodOp[n - 1];

the value of the most significant bit of the current state, namely CurrState[j - 1];

the value of the other j - 1 bits of the current state, namely CurrState[j - 2 : 0]; and

the value of the output signal Sel.

The diagram of Figure 5 represents the structure of the memory of the control unit UC1, where, at the address 0, there appears the idle state IDLE. The control unit UC1 remains in this state until there arrives at input an operating code and the inputs such that said unit is involved in the generation of output control signals.

In the aforesaid idle state, the respective output lines corresponding to the signal CS1 are kept at "0." In the table of Figure 5, there are comprised altogether 2^{n+k+j} allowed states for the possible machine. Each state is represented by a sequence of j + m bits, in which the first j bits (NS1) identify the next state, whilst the last n bits (CS1) correspond to the corresponding output signal.

In the unit UC1, the number j of state bits is preferably greater than or equal to the number of state bits of the control unit UC0. This makes possible, in

the decoding step, total coverage of the states of the control unit UC0 by the control unit UC1.

The proposed solution enables duplication of the instruction set of a CPU simply by programming appropriately the programmable control unit.

The said solution likewise enables execution of the operation of debugging of the non-programmed control unit, *i.e.*, the unit UC0, with the possibility, in the case where there arise problems on one or more instructions, of deciding to implement the said instructions using the programmable unit UC1.

The same solution also determines an increase in the accessibility of the internal nodes for debugging purposes, likewise enabling generation, for a given instruction, of control signals different from the ones generated by the non-programmable control unit.

Furthermore, the solution also leads to a reduction in the costs of implementation of complex instructions.

All of the above U.S. patents, U.S. patent application publications, U.S. patent applications, foreign patents, foreign patent applications and non-patent publications referred to in this specification and/or listed in the Application Data Sheet, are incorporated herein by reference, in their entirety.

Of course, without prejudice to the principle of the invention, the details of implementation and the embodiments may be amply varied with respect to what is described and illustrated herein, purely by way of non-limiting example, without thereby departing from the scope of the present invention, as defined in the claims that follow.